Knight Unlimited
Unholy Crusades

Groupe 1 – Mars 2024

Intelligence Artificielle

```
private bool shouldTurn()
{
    if (Side.IsColliding())
        return true;
    if (moveDir == 1)
        return !Down[1].IsColliding();
    if (moveDir == -1)
        return !Down[0].IsColliding();
    return false;
}
```

```
public override void _Ready()
{
    velocity = Vector2.Zero;
    velocity.X = speed;
    area_right = GetNode<Area2D>(path: "Area2DRight");
    area_left = GetNode<Area2D>(path: "Area2DLeft");
    animation = GetNode<AnimatedSprite2D>(path: "AnimatedSprite2D");
    player = (Playeru)GetParent().GetParent().FindChild(pattern:"Player(no animation tree)");
    Down[0] = GetNode<RayCast2D>(path: "RayCast2DDownLeft");
    Down[1] = GetNode<RayCast2D>(path: "RayCast2DDownRight");
    Side = GetNode<RayCast2D>(path: "RayCast2DSide");
    Parameters = new EntityHandler(hp:30, attack:4, armor:2, speed:250, currentWeapon:null, currentArmor:null);
}
```

```
public override void _PhysicsProcess(double delta)
{
    frames = delta;
    velocity = Velocity;
    if (shouldTurn())
        moveDir *= -1;
    if (moveDir == -1)
    {
        animation.FlipH = true;
    }
    else
    {
        animation.FlipH = false;
    }
    if (animation.FlipH)
    {
        area_right.Monitoring = false;
        area_left.Monitoring = true;
        Side.TargetPosition = new Vector2( x: -400,  y: 0);
    }
```

```
    else
    {
        area_right.Monitoring = true;
        area_left.Monitoring = false;
        Side.TargetPosition = new Vector2( x: 400,  y: 0);
    }
    if (!animationlock)
    {
        animation.Play( name: "default");
        velocity.X = speed * moveDir;
    }
    else
    {
        velocity.X = 0;
    }
    velocity.Y += gravity * (float)delta;
    Velocity = velocity;
    MoveAndSlide();
}
```

Jouabilité

```
public partial class Eldric : Playeru
{
    public Dictionary<SkillType, Skill> MeleeSkills { get; set; }
    public Dictionary<SkillType, Skill> RangedSkills { get; set; }
    1 usage  ZeDodongo
    public Eldric()
    {
        Parameters = new EntityComponent( inventory: new List<Resource>(), baseStats: new Dictionary<StatType, Stat>(){{StatType.HitPoints, new HitPoints( amount: 80)},{StatType.Attack, new Attack( amount: 10)},
    }
    1+3 usages  ZeDodongo
    public override void _Ready()
    {
        base._Ready();
    }

    1+3 usages  ZeDodongo
    public override void _PhysicsProcess(double delta)
    {
        base._PhysicsProcess(delta);
    }
}
```

```dotnet
public override void _PhysicsProcess(double delta)
{
    if(state == "default" || state == "attacking")
    {
        velocity = Velocity;
        // Add the gravity.
        if (Input.IsActionJustPressed("Attack") == (state != "attacking"))
        {
            attack();
        }
        if (!IsOnFloor() && animation.Animation != "attacking")
        {
            lock_anim = true;
            velocity.Y += gravity * (float)delta;
            if (velocity.Y < -10)
                animation.Play("jump");
            else if (velocity.Y > 10)
                animation.Play("fall");
            else
                animation.Play("jump_to_fall");
        }
        else if (IsOnFloor())
            lock_anim = false;


        // Handle Jump.
        if (Input.IsActionJustPressed("jump") && IsOnFloor())
        {
            lock_anim = true;
            Jump();
        }
        if (Input.IsActionJustPressed("jump") && !IsOnFloor())
        {
            lock_anim = true;
            DoubleJump();
            doubleJump = true;
        }
        if (IsOnFloor())
        {
            doubleJump = false;
        }

        // Get the input direction and handle the movement/deceleration.
        // As good practice, you should replace UI actions with custom gameplay actions.
        Vector2 direction = Input.GetVector("move_left", "move_right", "move_up", "move_down");
        if (direction != Vector2.Zero)
        {
            velocity.X = direction.X * Speed;
        }
        else
        {
            velocity.X = Mathf.MoveToward(Velocity.X, 0, Speed);
        }
```

```dotnet
            Velocity = velocity;
            update();
            MoveAndSlide();
            for (int i = 0; i < GetSlideCollisionCount(); i++)
            {
                var collision = GetSlideCollision(i);
                var a = collision.GetCollider();
                if (a.GetType() == typeof(enemy))
                {
                    TakeDamage(delta,(enemy)a);
                    animation.Play("hurt");
                    Velocity = velocity;
                    MoveAndSlide();
                }
            }
        }
        else if (state == "damaged")
        {
            velocity.Y += gravity * (float)delta;
            Velocity = velocity;
            MoveAndSlide();
            animation.Play("hurt");
            if(IsOnFloor())
                state = "default";
        }
    }
}
```

```
public EntityComponent(List<Resource> inventory, Dictionary<StatType, Stat> baseStats, Weapon? currentWeapon, Armor? currentArmor)
{
    CurrentStats = new Dictionary<StatType, Stat>() {{ StatType.Attack, new Attack( amount: 0) }, { StatType.HitPoints, new HitPoints( amount: 0) }, { StatType.NaturalArmor, new NaturalArmor( amount: 0) }, { StatType.Speed, new Speed( amount: 0)}};
    Inventory = inventory;
    BaseStats = baseStats;
    CurrentWeapon = currentWeapon;
    CurrentArmor = currentArmor;
    BonusStats = new Dictionary<StatType, Stat>() {{ StatType.Attack, new Attack( amount: 0) }, { StatType.HitPoints, new HitPoints( amount: 0) }, { StatType.NaturalArmor, new NaturalArmor( amount: 0) }, { StatType.Speed, new Speed( amount: 0)}};
    if (CurrentArmor != null)
    {
        foreach (var stat in CurrentArmor.GivenStats)
        {
            BonusStats[stat.Type].Amount += stat.Amount;
        }
    }
    if (CurrentWeapon != null)
    {
        foreach (var stat in CurrentWeapon.GivenStats)
        {
            BonusStats[stat.Type].Amount += stat.Amount;
        }
    }
    UpdateStats();
}

3 usages   ZeDodongo
public void UpdateStats()
{
    foreach (var stat KeyValuePair<StatType,Stat> in BaseStats)
    {
        CurrentStats[stat.Key].Amount = stat.Value.Amount + BonusStats[stat.Key].Amount;
    }
}

1 usage   ZeDodongo +1
public void UnequipWeapon()
{
    if (CurrentArmor == null)
        return;
    foreach (var stat in CurrentWeapon.GivenStats)
    {
        BonusStats[stat.Type].Amount -= stat.Amount;
    }

    Inventory.Add(CurrentWeapon);
}
```